Naïve Bayes Classifiers

Review

- Let event D = data we have observed.
- Let events $H_1, \ldots, H_n$ be events representing the *n* hypotheses we want to choose between.
- Use D to pick the "best" H.

- There are two "standard" ways to do this, depending on what information we have available.

Maximum likelihood hypothesis

- The maximum likelihood hypothesis (H^{ML}) is the hypothesis that **maximizes the probability of the data given that hypothesis.**

$$H^{\mathrm{ML}} = \underset{i}{\operatorname{argmax}}\, P(D \mid H_i)$$

- How to use it: compute $P(D \mid H_i)$ for each hypothesis (1 through *n*) and select the one with the greatest value.

Maximum a posteriori (MAP) hypothesis

- The MAP hypothesis is the hypothesis that **maximizes the posterior probability**:

$$\begin{aligned} H^{\mathrm{MAP}} &= \operatorname*{argmax}_i P(H_i \mid D) \\ &= \operatorname*{argmax}_i \frac{P(D \mid H_i)P(H_i)}{P(D)} \\ &\propto \operatorname*{argmax}_i P(D \mid H_i)P(H_i) \end{aligned}$$

- The $P(D \mid H_i)$ terms are now *weighted* by the hypothesis prior probabilities.

Posterior probability

- If you need the actual posterior probability for some hypothesis H_i:

$$P(H_i \mid D) = \frac{P(D \mid H_i)P(H_i)}{P(D)}$$

$$= \frac{P(D \mid H_i)P(H_i)}{\sum_j P(D, H_j)}$$

$$= \frac{P(D \mid H_i)P(H_i)}{\sum_j P(D \mid H_j)P(H_j)}$$

Combining evidence

- If we have multiple pieces of data/evidence (say two pieces), then we need to compute or estimate

$$P(D_1, D_2 \mid H_i)$$

which is often hard.

- Instead, we **assume** all pieces of evidence are conditionally independent given a hypothesis:

$$P(D_1, D_2 \mid H_i) = P(D_1 \mid H_i)P(D_2 \mid H_i)$$

- This assumption is **most likely not true**, but we do it to make our lives easier.

Combining evidence (*m* pieces)

$$P(H_i \mid D_1, \ldots, D_m) = \frac{P(D_1, \ldots, D_m \mid H_i) P(H_i)}{P(D_1, \ldots, D_m)}$$

$$= \frac{\Big[P(D_1 \mid H_i) \cdots P(D_m \mid H_i)\Big] P(H_i)}{P(D_1, \ldots, D_m)}$$

$$= \frac{\Big[\prod_{j=1}^{m} P(D_j \mid H_i)\Big] P(H_i)}{P(D_1, \ldots, D_m)}$$

where

$$P(D_1 \ldots, D_m) = \sum_{k=1}^{n} \left(\Big[\prod_{j=1}^{m} P(D_j \mid H_k) \Big] P(H_k) \right)$$

Classification

- Classification is the problem of identifying which of a set categories (called classes) a particular item belongs in.
- Lots of real-world problems are classification problems:
  - spam filtering (classes: spam/not-spam)
  - handwriting recognition & OCR (classes: one for each letter, number, or symbol)
  - text classification, image classification, music classification, etc.
- Almost any problem where you are assigning a label to items can be set up as a classification task.

Classification

- An algorithm that does classification is called a classifier. Classifiers take an item as input and output the class it thinks that item belongs to. That is, the classifier *predicts* a class for each item.
- Lots of classifiers are based on probabilities and statistical inference:
  - The **classes** become the hypotheses being tested.
  - The item being classified is turned into a collection of data called **features**. Useful features are attributes of the item that are strongly correlated with certain classes.
  - The classification algorithm is usually ML or MAP, depending on what data we have available.

Example: Spam classification

- New email arrives: is it spam or not spam?
- A useful set of features might be the presence or absence of various words in the email:
  - F1, ~F1: "Kirlin" appears/does not appear
  - F2, ~F2: "viagra" appears/does not appear
  - F3, ~F3: "cash" appears/does not appear
- Let's say our new email contains "Kirlin" and "cash," but not "viagra."
- The features for this email are F1, ~F2, and F3.
- Let's use MAP for classification.

Example: Spam classification

- Features = Data = D = F1, ~F2, F3.

$$H^{\mathrm{MAP}} = \operatorname*{argmax}_{i} P(D \mid H_i) P(H_i)$$

$$H^{\mathrm{MAP}} = \operatorname*{argmax}_{i \in \{\text{spam}, \text{not-spam}\}} P(F_1, \neg F_2, F_3 \mid H_i) P(H_i)$$

Learning probabilities from data

- To use MAP, we need to calculate or estimate P(Hi) and P(F1, ~F2, F3 | Hi) for each i.
- In other words, we need to know:
  - P(spam)
  - P(not-spam)
  - P(F1, ~F2, F3 | spam)
  - P(F1, ~F2, F3 | not-spam)

Learning probabilities from data

- Let's assume we have access to a large number of old emails that are correctly labeled as spam/not-spam.
- How can we estimate P(spam)?

$$P(\text{spam}) = \frac{\#\text{ of emails labeled as spam}}{\text{total }\#\text{ of emails}}$$

Learning probabilities from data

- Let's assume we have access to a large number of old emails that are correctly labeled as spam/not-spam.
- How can we estimate P(F1, ~F2, F3 | spam)?

$$P(F_1, \neg F_2, F3 \mid \text{spam}) = \frac{\text{\# of spam emails with those exact features}}{\text{total \# of spam emails}}$$

- Why is this probably going to be a very rough estimate?

Conditional independence to the rescue!

- It is unlikely that our set of old emails contains many messages with that exact set of features.
- Let's make an assumption that all of our features are conditionally independent of each other, given the hypothesis (spam/not-spam).

$$P(F_1, \neg F_2, F3 \mid \text{spam}) =$$

$$P(F_1 \mid \text{spam}) \cdot P(\neg F_2 \mid \text{spam}) \cdot P(F_3 \mid \text{spam})$$

- These probabilities are easier to get good estimates for!
- A classifier that makes this assumption is called a Naïve Bayes classifier.

Learning probabilities from data

- So now we need to estimate P(F1 | spam) instead of P(F1, ~F2, F3 | spam).
- Equivalently, how can we estimate the probability of seeing "Kirlin" in an email given that the email is spam?

$$P(F_1 \mid \text{spam}) = \frac{\#\text{ of spam emails with the word Kirlin}}{\text{total }\#\text{ of spam emails}}$$

Example

Suppose I know that 80% of my email is spam. I have 3 features: `luxury`, `brands`, *and* `save`. *For each email, I will therefore have 3 pieces of data—the presence or absence of each one of these features. I know* $\mathbb{P}[\text{luxury} \mid \text{spam}] = 0.4$ *and* $\mathbb{P}[\text{brands} \mid \text{spam}] = 0.3$ *and* $\mathbb{P}[\text{save} \mid \text{spam}] = 0.4$ *and* $\mathbb{P}[\text{luxury} \mid \text{not spam}] = 0.01$ *and* $\mathbb{P}[\text{brands} \mid \text{not spam}] = 0.2$ *and* $\mathbb{P}[\text{save} \mid \text{not spam}] = 0.1$. *Suppose an email includes* `luxury` *and* `save` *but not* `brands` . *Should it be classified as spam or not spam?*

Another problem to handle…

- What if we see a word we've never encountered before? What happens to its probability estimate? (and why is this bad?)

$$P(F_j \mid \text{spam}) = \frac{\text{\# of spam emails with word } F_j}{\text{total \# of spam emails}}$$

$$P(\text{spam} \mid F1, \ldots, F_m) = \frac{\left[\prod_{j=1}^{m} P(F_j \mid \text{spam})\right] P(\text{spam})}{P(F_1, \ldots, F_m)}$$

- Probability of zero destroys the entire calculation!

Another problem to handle…

- Fix the estimates:

$$P(F_j \mid \text{spam}) = \frac{\text{\# of spam emails with word } F_j + 1}{\text{total \# of spam emails} + 2}$$

- This is called ***smoothing***. Removes the possibility of a zero probability wiping out the entire calculation.
- Simulates adding two additional spam emails, one containing every word, and containing no words.
  – We would also smooth for non-spam: adding two non-spam emails, one with all words, one with no words.

Summary of Naïve Bayes

- Assumes the data is a collection of features, and each feature is conditionally independent of all other features given the hypothesis.
- Classifies using MAP hypothesis.

Summary of Naïve Bayes

- Hypotheses: H_1 through H_n.
- Features (data): F_1 through F_m.

$$
\begin{aligned}
H^{\mathrm{MAP}} &= \operatorname*{argmax}_i P(D \mid H_i) P(H_i) \\
&= \operatorname*{argmax}_i P(F_1, \ldots, F_m \mid H_i) P(H_i) \\
&= \operatorname*{argmax}_i \Big[P(F_1 \mid H_i) \cdots P(F_m \mid H_i) \Big] P(H_i) \\
&= \operatorname*{argmax}_i \Big[\prod_{j=1}^{m} P(F_j \mid H_i) \Big] P(H_i)
\end{aligned}
$$

Summary of Naïve Bayes

- Probabilities needed to be determined (either given to you or estimated from data):

- P(H_i) for i = 1 to n.
- P(F_j | H_i) for j = 1 to m and i = 1 to n.

Summary of Naïve Bayes (for email)

- Naïve Bayes classifies using MAP:

$$
\begin{aligned}
H^{\mathrm{MAP}} &= \operatorname*{argmax}_i P(D \mid H_i) P(H_i) \\
&= \operatorname*{argmax}_{i \in \{\text{spam}, \text{not-spam}\}} P(F_1, \ldots, F_m \mid H_i) P(H_i) \\
&= \operatorname*{argmax}_{i \in \{\text{spam}, \text{not-spam}\}} \Big[P(F_1 \mid H_i) \cdots P(F_m \mid H_i) \Big] P(H_i) \\
&= \operatorname*{argmax}_{i \in \{\text{spam}, \text{not-spam}\}} \Big[\prod_{j=1}^{m} P(F_j \mid H_i) \Big] P(H_i)
\end{aligned}
$$

- Compute this for spam and for not-spam; see which is bigger.

Summary of Naïve Bayes (for email)

- Estimating the ***prior*** for each hypothesis:

$$P(H_i) = \frac{\#\text{ of emails labeled as } H_i}{\text{total } \#\text{ of emails}}$$

- Estimating the probability of a feature given a class (aka ***likelihood***):

$$P(F_j \mid H_i) = \frac{\#\text{ of } H_i \text{ emails with word } F_j + 1}{\text{total } \#\text{ of } H_i \text{ emails} + 2}$$